PCT WELTORGANISATION FÜR GEISTIGES EIGENTUM
Internationales Büro

INTERNATIONALE ANMELDUNG VERÖFFENTLICHT NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT)

(51) Internationale Patentklassifikation 6 : G06F 11/00	A1	(11) Internationale Veröffentlichungsnummer: **WO 00/00890** (43) Internationales Veröffentlichungsdatum: 6. Januar 2000 (06.01.00)

(21) Internationales Aktenzeichen: PCT/EP99/04438

(22) Internationales Anmeldedatum: 25. Juni 1999 (25.06.99)

(30) Prioritätsdaten:
198 30 015.8 26. Juni 1998 (26.06.98) DE

(71) Anmelder *(für alle Bestimmungsstaaten ausser US)*: DEUTSCHE TELEKOM AG [DE/DE]; Friedrich–Ebert–Allee 140, D–53113 Bonn (DE).

(72) Erfinder; und
(75) Erfinder/Anmelder *(nur für US)*: POSEGGA, Joachim [DE/DE]; Eichelbergweg 16A, D–76646 Bruchsal (DE). VOGT, Harald [DE/DE]; Liebfrauenstrasse 5, D–64289 Darmstadt (DE).

(74) Gemeinsamer Vertreter: DEUTSCHE TELEKOM AG; Rechtsabteilung (Patente) PA1, D–64307 Darmstadt (DE).

(81) Bestimmungsstaaten: US, europäisches Patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Veröffentlicht
Mit internationalem Recherchenbericht.
Vor Ablauf der für Änderungen der Ansprüche zugelassenen Frist; Veröffentlichung wird wiederholt falls Änderungen eintreffen.

(54) Title: METHOD FOR CHECKING JAVA BYTE CODE PROGRAMMES FOR SECURITY CHARACTERISTICS

(54) Bezeichnung: VERFAHREN ZUR PRÜFUNG VON JAVA–BYTECODE–PROGRAMMEN AUF SICHERHEITSEIGENSCHAFTEN

(57) Abstract

The invention relates to a method for checking Java byte code programmes for security characteristics. The technical aim of the invention is to provide a method for guaranteeing the best possible security in checking the security characteristics of byte code programmes. According to the invention, the mode of operation of the byte code programme being checked is configured for a finite status transition system (M) and the state space of the JVM is configured for a finite quantity of states in M. After being entered into a model checker, the data of the status transition system (M) is compared with the data in the model checker, the data in the model checker having been entered as a set of conditions (S) for the characteristics of a reliable byte code programme. The byte code programme being checked is only released for further processing if the status transition system (M) fulfils all of the conditions of the set (S). The invention therefore provides a means of guaranteeing the security of byte code programmes and with additional enhancements, can guarantee a certain functionality. This increases the reliability of applications which are run on security–critical platforms such as smart cards.

(57) Zusammenfassung

Die technische Aufgabe ist auf ein Verfahren ausgerichtet, das eine höchstmögliche Sicherheit bei der Überprüfung von Sicherheitseigenschaften von Bytecode–Programmen gewährleistet. Erfindungsgemäß wird die Funktionsweise des zu prüfenden Bytecode–Programms auf ein endliches Zustandsübergangssystem (M) und der Zustandsraum der JVM auf eine endliche Menge von Zuständen in M ausgebildet. Nach Eingabe in einen Modelchecker werden die Daten des endlichen Zustandsübergangssystem (M) mit den im Modelchecker als Bedingungsmenge (S) eingegebenen Daten für Eigenschaften die ein zulässiges Bytecode–Programm kennzeichnen, verglichen. Das zu überprüfende Bytecode–Programm wird nur zur weiteren Verarbeitung freigegeben, wenn das Zustandsübergangssystem (M) alle Bedingungen der Bedingungsmenge (S) erfüllt. Durch die beschriebene Technik kann die Sicherheit von Bytecode–Programmen garantiert und durch zusätzliche Erweiterungen eine gewisse Funktionalität zugesichert werden. Damit kann das Vertrauen in Anwendungen erhöht werden, die auf sicherheitskritischen Plattformen wie Smartcards ablaufen sollen.

LEDIGLICH ZUR INFORMATION

Codes zur Identifizierung von PCT-Vertragsstaaten auf den Kopfbögen der Schriften, die internationale Anmeldungen gemäss dem PCT veröffentlichen.

AL	Albanien	ES	Spanien	LS	Lesotho	SI	Slowenien
AM	Armenien	FI	Finnland	LT	Litauen	SK	Slowakei
AT	Österreich	FR	Frankreich	LU	Luxemburg	SN	Senegal
AU	Australien	GA	Gabun	LV	Lettland	SZ	Swasiland
AZ	Aserbaidschan	GB	Vereinigtes Königreich	MC	Monaco	TD	Tschad
BA	Bosnien-Herzegowina	GE	Georgien	MD	Republik Moldau	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagaskar	TJ	Tadschikistan
BE	Belgien	GN	Guinea	MK	Die ehemalige jugoslawische Republik Mazedonien	TM	Turkmenistan
BF	Burkina Faso	GR	Griechenland			TR	Türkei
BG	Bulgarien	HU	Ungarn	ML	Mali	TT	Trinidad und Tobago
BJ	Benin	IE	Irland	MN	Mongolei	UA	Ukraine
BR	Brasilien	IL	Israel	MR	Mauretanien	UG	Uganda
BY	Belarus	IS	Island	MW	Malawi	US	Vereinigte Staaten von Amerika
CA	Kanada	IT	Italien	MX	Mexiko		
CF	Zentralafrikanische Republik	JP	Japan	NE	Niger	UZ	Usbekistan
CG	Kongo	KE	Kenia	NL	Niederlande	VN	Vietnam
CH	Schweiz	KG	Kirgisistan	NO	Norwegen	YU	Jugoslawien
CI	Côte d'Ivoire	KP	Demokratische Volksrepublik Korea	NZ	Neuseeland	ZW	Zimbabwe
CM	Kamerun			PL	Polen		
CN	China	KR	Republik Korea	PT	Portugal		
CU	Kuba	KZ	Kasachstan	RO	Rumänien		
CZ	Tschechische Republik	LC	St. Lucia	RU	Russische Föderation		
DE	Deutschland	LI	Liechtenstein	SD	Sudan		
DK	Dänemark	LK	Sri Lanka	SE	Schweden		
EE	Estland	LR	Liberia	SG	Singapur		

Verfahren zur Prüfung von Java-Bytecode-Programmen auf Sicherheitseigenschaften

Beschreibung:

Die Erfindung betrifft ein Verfahren zur Prüfung von Java-Bytecode-Programmen auf Sicherheitseigenschaften nach dem Patentanspruch 1.

Java ist eine von Sun Microsystems entwickelte Programmiersprache, die in sog. Bytecode-Programme übersetzt wird. Obwohl ursprünglich für Java entworfen, eignet sich Bytecode auch als Zielsprache für andere Programmiersprachen, und es gibt bereits entsprechende Compiler (z. B. für Ada).
Das besondere Konzept von Java besteht darin, Programme im Netz abzulegen und von Netzteilnehmern abrufen zu lassen, um sie auf einem Gerät des Netzteilnehmers ablaufen zu lassen. Dies unterstützt z. B. die Konfiguration und Wartung von Geräten aus der Ferne, ist aber auch im Zusammenhang mit Smartcards interessant, deren Funktionalität sich auf diese Weise erweitern läßt. Bytecode-Programme haben folgende Eigenschaften, die für ihren Einsatzzweck bedeutend sind:

- Bytecode-Programme sind kompakt. Bytecode-Instruktionen bieten wesentlich mehr Funktionalität als z. B. Instruktionen von Maschinensprachen. Durch die Anlehnung an Java werden Konzepte wie Objektorientierung direkt unterstützt.
- Bytecode-Programme können effizient interpretiert werden. Dies verleiht solchen Programmen Unabhängigkeit von einer bestimmten Zielmaschine. Vorausgesetzt wird lediglich ein Interpreter (die sog. "Java virtual machine", JVM) auf der Zielmaschine, um beliebige Bytecode-Programme ausführen zu können. Dieser Interpreter kann selbst kompakt implementiert und in (fast) jedes Gerät integriert werden.

Die JVM interpretiert eine Eingabe in Form einer Folge von Zeichen als Bytecode-Programm. Die JVM nimmt dabei keine Prüfungen vor, ob es sich bei der Zeichenfolge tatsächlich um ein Bytecode-Programm handelt. Sie verarbeitet die Zeichen als Bytecode-Befehle, ohne eine genauere Prüfung auf ihre Korrektheit durchzuführen. Die JVM verzichtet auf eine Überprüfung des Bytecodes, da sonst die Ausführungsgeschwindigkeit erheblich leiden würde. Prinzipiell ist es also möglich, eine beliebige Folge von Zeichen als

Bytecode-Programm interpretieren zu lassen. Dadurch wäre es auch möglich - bei genauerer Kenntniss der Implementierung der JVM - ihre Sicherheitsmechanismen zu umgehen und damit z. B. auf dem ausführenden Rechner Daten auszuspähen. Ein sicheres Bytecode-Programm kann dagegen aufgrund des Sprachentwurfs und der JVM auf Daten und Ressourcen des Zielrechners nur zugreifen, indem es dafür Funktionen der JVM in Anspruch nimmt.

Um zu verhindern, daß Zeichenfolgen verarbeitet werden, die keine zulässigen (sicheren) Bytecode-Programme darstellen, ist der JVM ein Prozeß vorgeschaltet, der eine Zeichenfolge auf Konformität zu gewissen Anforderungen überprüft, die sog. Bytecode-Verifikation. Diese Anforderungen garantieren, daß eine Zeichenfolge ein sicheres Bytecode-Programm darstellt. Da diese Überprüfung nur einmal vorgenommen werden muß, entstehen zur Laufzeit keine Nachteile bzgl. der Ausführungsgeschwindigkeit. Der Prozeß der Bytecode-Verifikation prüft eine Folge von Zeichen daraufhin, ob sie ein zulässiges Bytecode-Programm darstellt und damit ohne Gefahr für die Integrität des ausführenden Geräts der JVM zur Interpretierung übergeben werden kann (siehe T. Lindholm, F. Yellin „The Java Virtual Machine Specification"; Sun Microsystems; Addison-Wesley, 1996.

Ein zulässiges Bytecode-Programm ist durch gewisse Eigenschaften gekennzeichnet, die in T. Lindholm, F. Yellin: The Java Virtual Machine Specification. Sun Microsystems, 1996 als "structural constraints" beschrieben werden. Im wesentlichen beschreiben sie die Typsicherheit eines Bytecode-Programms, d. h. in einem Bytecode-Programm sind die Instruktionen so angeordnet, daß eine Instruktion stets solche Daten als Parameter bekommt, die ihrem Typ nach dem entsprechen, was die Instruktion erwartet. Diese Eigenschaften stellen sicher, daß das Bytecode-Programm unter Kontrolle der JVM bleibt und z. B. nicht direkt auf Rechnerressourcen zugreifen kann.

Der Prozeß der Bytecode-Verifikation ist durch zwei Gegebenheiten beschrieben: erstens durch die Beschreibung der Eigenschaften, die er für ein Bytecode-Programm zusichern soll; zweitens durch die Implementierung des Prozesses (dabei handelt es sich um ein C-Programm). Beide Beschreibungen eignen sich nicht, um formale, maschinelle

Berechnungen durchzuführen und damit die Sicherheit von Bytecode-Programmen formal zu garantieren.

Weiterhin ist ein mit Modelchecker bezeichnetes Werkzeug zur Untersuchung von Zustandsübergangssystemen bekannt (siehe K. L. Mc Millan „Symbolic Model Cheking" Kluwer Academic Publishers, 1993).

Zustandsübergangssysteme sind ein allgemeines Modell für Programme, die auf Rechnern oder anderen Maschinen (wie der JVM) ablaufen. Dabei betrachtet man die Zustände, die die Maschine während der Ausführung des Programmes einnimmt und welche Übergänge, d. h. Zustandsänderungen, dabei möglich sind.

Modelchecking ist eine Technik, mit der nur endliche Zustandsübergangssysteme untersucht werden können. Ein Modelchecker untersucht dafür den gesamten Zustandsraum des Systems, um Eigenschaften des Systems zu berechnen. Eine verbreitete Anwendung von Modelcheckern ist, ein System daraufhin zu prüfen, ob es gewisse Eigenschaften erfüllt. Der Modelchecker benötigt dafür als Eingabe eine Beschreibung des Systems, das untersucht werden soll sowie die Beschreibung von Eigenschaften, deren Gültigkeit im System festgestellt werden soll. In beiden Fällen handelt es sich um formale Beschreibungen, d. h. Beschreibungen, deren formale Semantik feststeht, und auf denen somit mathematische Berechnungen durchgeführt werden können.

Im allgemeinen besitzen Software-Systeme auch Bytecode-Programme einen unendlichen Zustandsraum. Das bedeutet, daß das Verfahren des Modelchecking auf Bytecode-programme, die durch einen unendlichen Zustandsraum gekennzeichnet sind, nicht anwendbar ist.

Die technische Aufgabe ist auf ein Verfahren ausgerichtet, das eine höchstmögliche Sicherheit bei der Überprüfung von Sicherheitseigenschaften von Bytecode-Programmen gewährleistet.

Ausgangspunkt des erfindungsgemäßen Verfahrens ist der Sachverhalt, daß ein Bytecode-Programm eine Folge von Zeichen (Bytes) beinhaltet, wobei jedes Zeichen entweder als Instruktion oder als Datum (als Eingabe für die vorhergehende Instruktion) interpretiert wird. Ein Bytecode-Programm stellt somit eine Folge von Instruktionen und zugehörigen

Daten dar. Ein Bytecode-Programm kann damit auch als Beschreibung eines Zustandsübergangssystems gedeutet werden, das Zustände der JVM transformiert. Die Zustände des Systems werden mit den Zuständen der JVM identifiziert und die Übergänge durch die Instruktionen des Programms festgelegt.

Erfindungsgemäß wird das oben beschriebene Zustandsübergangssystem (mit potentiell unendlichem Zustandsraum) durch eine geeignete Abbildung auf ein System mit einer endlichen Anzahl von Zuständen abgebildet. Diese Abstraktion auf eine endliche Anzahl von Zuständen wird dadurch erreicht, daß das System alle Informationen verwirft, die nicht benötigt werden, um festzustellen, ob das ursprüngliche Bytecode-Programm zulässig ist. Dies geschieht im wesentlichen dadurch, daß die konkreten Datenwerte durch reine Typinformationen ersetzt werden. Der Ersatz der konkreten Datenwerte durch reine Typinformationen ist möglich, weil die Zulässigkeit eines Bytecode-Programms nicht von konkreten Werten abhängt. Dabei bleiben aber alle Informationen, die notwendig sind, um die Zulässigkeit des Bytecode-Programms nachzuweisen, erhalten. Das resultierende Zustandsübergangssystem M besitzt eine endliche Menge von Zuständen und erfüllt damit die Grundbedingung für die Bearbeitung in einem Modelchecker. Das resultierende Zustandsübergangssystem M wird in den Modelchecker eingegeben.
In einem weiteren Schritt werden die Eigenschaften, die wie in T. Lindholm, F. Yellin: The Java Virtual Machine Specification. Sun Microsystems, 1996 als "structural constraints" beschrieben, ein zulässiges Bytecode-Programm kennzeichnen, in einer Logik formuliert, die der Modelchecker als Spezifikationssprache für Systemeigenschaften versteht. Das Ergebnis ist eine Menge von Formeln, die dem Modelchecker als Bedingungsmenge S als Vergleichsbasis für das Zustandsübergangssystem M übergeben werden. Aufgrund der o.g. Verfahrensweise ist gewährleistet, daß sich die Formeln nur auf Informationen beziehen, die sowohl im ursprünglichen, potentiell zustandsunendlichen System, als auch im daraus gewonnenen zustandsendlichen System vorhanden sind.

Der Modelchecker wird mit den eben beschriebenen Eingaben gestartet. Als Ergebnis liefert er die Information, ob das Zustandsübergangssystem M die in der Spezifikationssprache als Bedingungsmenge S in Form von Formeln beschriebenen Systemeigenschaften erfüllt. Dabei prüft der Modelchecker für jede Bedingung s der Bedingungsmenge S, ob sie vom

Zustandsübergangssystem M des zu prüfenden Bytecode-Programms erfüllt ist. Wenn das der Fall ist, so ist sichergestellt, daß es sich bei dem ursprünglichen Bytecode-Programm um ein zulässiges Programm handelt, das die Sicherheit des ausführenden Rechners nicht bedroht. Das Bytecode-Programm wird zur weiteren Bearbeitung durch den Rechner freigegeben.

Erfüllen die im Zustandsübergangssystem M erfaßten Daten des zu prüfenden Bytecode-Programms jedoch nicht die in der Bedingungsmenge S in Form von Formeln beschriebenen Systemeigenschaften, die ein Bytecode-Programm kennzeichnen, ist grundsätzlich davon auszugehen, daß das geprüfte Bytecode-Programm die Sicherheit des Rechners bedrohen kann. Das geprüfte Bytecode-Programm wird nicht zur weiteren Bearbeitung freigegeben..

Das erfindungsgemäße Verfahren wird nachfolgend näher erläutert:

Zweck der "Bytecode-Verifikation" ist es, eine Klassendatei, also die Einheit, in der eine Java-Klassenbeschreibung zusammengefaßt ist, auf sichere Ausführbarkeit zu prüfen. Dazu werden die einzelnen Methoden der Klasse (hier Bytecode-Programme genannt) einzeln herausgelöst und der eigentlichen Bytecode-Verifikation unterzogen.

Eine Klassendatei enthält zusätzliche Daten, im wesentlichen Konstanten, auf die in den Bytecode-Programmen Bezug genommen wird. Für die folgenden Schritte ist es vorteilhaft, diese Referenzen in den Bytecode-Programmen aufzulösen und in die Programme einzuarbeiten, bevor mit der eigentlichen Prüfung begonnen wird.

Durch diese Vorverarbeitung erhält man eine Beschreibung der Bytecode-Programme, die sich nicht wesentlich von der ursprünglichen Form unterscheidet.

Die JVM, der Standard-Interpreter für Bytecode-Programme, ist durch eine abstrakte Maschine, d.h. eine Menge von Zuständen, beschrieben. Die Ausführung eines Bytecode-Programms entspricht der Interpretation von Bytecode-Instruktionen durch Zustandsübergänge. Eine Bytecode-Instruktion bewirkt dabei eine Transformation des aktuellen Zustands der JVM in einen neuen Zustand.

Ein Bytecode-Programm definiert also ein Zustandsübergangssystem, wobei der Zustandsraum durch die JVM festgelegt ist und die Übergänge durch die Instruktionen des

Bytecode-Programms bestimmt werden. Dieses Zustandsübergangssystem besitzt einen potentiell unendlichen Zustandsraum. Deshalb ist es nicht geeignet, um von einem Modelchecker untersucht zu werden. Dazu muß es auf ein endliches Zustandsübergangssystem M abgebildet werden. Diese Abbildung wird im folgenden beschrieben.

Es wird nicht zunächst ein unendliches Übergangssystem konstruiert, das anschließend auf ein endliches abgebildet wird. Vielmehr wird aus dem Bytecode-Programm direkt ein endliches Zustandsübergangssystem M konstruiert, unter Verwendung von Regeln, die das Verhalten der Bytecode-Instruktionen beschreiben.
Die Funktionsweise eines Bytecode-Programms wird abgebildet auf ein endliches Zustandsübergangssystem M. Der Zustandsraum der JVM wird auf eine endliche Menge von Zuständen im Zustandsübergangssystem M abgebildet. Die Bytecode-Instruktionen bewirken dann Übergänge auf dieser endlichen Zustandsmenge, wobei ihre prinzipielle Wirkung erhalten bleibt. Das Zustandsübergangssystem M wird so beschrieben, daß diese Beschreibung als Eingabe für den Modelchecker dienen kann.

Die Eingabe für den Modelchecker besteht aus zwei Teilen:

1. Einer als Zustandsübergangssystem M bezeichneten Systembeschreibung, bestehend aus
   - einer Beschreibung des Zustandsraums, festgelegt durch die Zustandsvariablen und ihre Wertebereiche;
   - einer Vorschrift, die die Startzustände des Systems festlegt;
   - einer Übergangsrelation, die angibt, wie Zustände transformiert werden und unter welchen Bedingungen.
2. Einer Menge von Spezifikationen, die als Bedingungsmenge S bezeichnet werden und die die gewünschte Eigenschaften eines zulässigen Bytecode-Programms beschreiben. Diese Eigenschaften werden von dem Zustandsübergangssystem M verlangt, das durch das Bytecode-Programm beschrieben wird. So kann die Sicherheit bei der Ausführung des Bytecode-Programms garantiert werden. Diese Eigenschaften gelten im konkreten, unendlichen Zustandsübergangssystem (und damit im ursprünglichen Bytecode-Programm) dann, wenn sie im abstrakten, endlichen Zustandsübergangssystem M gelten. Der Modelchecker prüft, ob sie im abstrakten Zustandsübergangssystem M

gelten. Falls dies der Fall ist, darf man schließen, daß die Eigenschaften auch vom konkreten System und damit im ursprünglichen Programm erfüllt werden.

Die Erzeugung der erforderlichen Daten wird im folgenden beschrieben:

Der Zustandsraum des Zustandsübergangssystems M ist aus folgenden Komponenten aufgebaut:

- Einem Befehlszähler.
- Einem Operanden-Stack, der die Parameter und Ergebnisse von Instruktionen aufnimmt.
- Einem Feld von lokalen Variablen.
- Einem Feld von globalen Variablen.
- Variablen für das Mitführen von buchhalterischen Informationen; auf diese kann zurückgegriffen werden, um Eigenschaften des Systems zu beschreiben, die mit den übrigen Komponenten nicht beschrieben werden können. Hierzu gehört u.a. ein
- Stack, der Informationen über die aktiven Subroutinen (Unterprogramme in einem Bytecode-Programm) enthält.

Die Werte von Variablen und Stackelementen stammen aus endlichen Wertebereichen. Der Befehlszähler kann eine endliche Zahl von Adresswerten annehmen. Der Operanden-Stack ist in der Höhe begrenzt. Insgesamt ist der Zustandsraum von M dadurch endlich.

Das Befehlsverzeichnis V enthält Beschreibungen aller Bytecode-Instruktionen, bestehend aus folgenden Informationen je Instruktion:

- Die Bezeichnung der Instruktion (1 Byte).
- Die Anzahl der Parameter in Bytes (Zeichen).
- Eine Beschreibung der Wirkung auf Zustände der JVM.
- Bedingungen, die ein Zustand der JVM erfüllen muß, damit die Instruktion angewendet werden kann (C1).
- Bedingungen (C2), die die JVM erfüllen muß, weil die Instruktion im Programm vorkommt. Diese Bedingungen beziehen sich nicht notwendigerweise auf einzelne Zustände, sondern auch auf Ausführungspfade, d.h. Folgen von Zuständen.

Für die Bedingungen in V gilt folgende Eigenschaft (A1): Die Beschreibung der Bedingungen im Befehlsverzeichnis V verwendet nur solche Symbole, die bei der Beschreibung des Zustandsraums für M verwendet werden.

Ein Übersetzer erzeugt aus einem Bytecode-Programm ein endliches Zustandsübergangssystem. Das Bytecode-Programm liegt als Folge von Zeichen in der Eingabe vor, wobei ein einzelnes Zeichen als Instruktion oder Parameter einer Instruktion gedeutet werden kann. Das erste Zeichen wird als Instruktion gedeutet.

Ziel der Übersetzung ist die Konstruktion eines Zustandsübergangssystems M und einer Bedingungsmenge S. Der Übersetzungsprozeß ist wie folgt beschrieben:

1. Beginne mit einem initialen ("leeren") Übergangssystem M. Der Zustandsraum von M ist schon festgelegt durch die abstrakte Repräsentation des JVM-Zustandsraums. Der Startzustand der JVM legt aber den Startzustand von M fest. Es sind jedoch noch keine Übergänge zwischen den Zuständen festgelegt.
2. Beginne mit einer leeren Bedingungsmenge S.
3. Solange Zeichen in der Eingabe vorhanden sind, führe folgende Schritte 4 bis 8 durch:
4. Lies ein Zeichen von der Eingabe; dieses bezeichnet die Programminstruktion B.
5. Schlage im Befehlsverzeichnis V nach, welche Parameter B benötigt.
6. Lies die für B benötigten Parameter P von der Eingabe.
7. Übergebe (B,P) an die Abstraktionskomponente.
8. Übergebe (B,P) an die Bedingungskomponente.

Die Abstraktionskomponente erzeugt aus einer Instruktion B und zugehörigen Parametern P eine Menge von Zustandsübergängen. Das Zustandsübergangssystem M wird um diese Übergänge erweitert.

1. Schlage im Befehlsverzeichnis V nach, welche Wirkung B auf einem Zustand der JVM hat. Die Wirkung ist durch eine Regel beschrieben, so daß sich der Folgezustand der JVM aus einer Berechnung auf dem aktuellen Zustand ergibt. (Eine explizite Angabe der möglichen Übergänge ist nicht möglich, da die JVM über einen (praktisch) unendlichen Zustandsraum verfügt.)
2. Erzeuge daraus eine Beschreibung der Wirkung auf Zustände von M unter Berücksichtigung von P. Die Parameter P gehen in die Berechnung des Folgezustands ein. Wendet man eine JVM-Regel auf einen M-Zustand an, so erhält man eine Menge von M-Zuständen, die die möglichen Folgezustände der JVM repräsentieren. Es ergibt sich eine Menge von Folgezuständen für M, da nicht die konkreten Werte in P, sondern

nur die für M relevanten Informationen betrachtet werden. Welcher JVM-Folgezustand bei Ausführung des Bytecode-Programms tatsächlich eingenommen wird, hängt von den konkreten Werten in P ab.

3. Diese Beschreibung legt eine Menge von Zustandsübergängen fest; erweitere M um diese Übergänge. Diese Übergänge können durch eine Regel beschrieben oder explizit angegeben werden. Letzteres ist möglich, da M nur eine endliche Zahl an Zuständen kennt und deshalb alle Übergänge durch Zustandspaare angegeben werden können. Letztlich lehnt sich die Beschreibung daran an, was der verwendete Modelchecker versteht. Aus praktischen Gründen wird man eine explizite Angabe aller Übergänge vermeiden, da die Datenmenge hierfür weit größer ist als bei der impliziten Angabe durch Regeln.

Die Bedingungskomponente erzeugt aus (B,P) eine Menge von Bedingungen, die vom Modelchecker letztlich zu prüfen sind. Diese Bedingungen können auf verschiedene Art repräsentiert werden, etwa durch temporallogische Formeln.

1. Schlage im Befehlsverzeichnis V nach, welche Bedingungen C1 an den aktuellen JVM-Zustand gestellt werden, damit B ausgeführt werden kann.
2. Übertrage C1 auf entsprechende M-Bedingungen. Laut (A1) sind die Bedingungen in V und die Zustandsbeschreibung für M so aufeinander abgestimmt, daß C1 auf M-Zuständen interpretiert werden kann. Die Übertragung auf M besteht im wesentlichen daraus, mittels P konkrete Instanzen von C1 zu erzeugen und in eine logische Formel zu übersetzen.
3. Schlage entsprechend die Bedingungen C2 nach, die für das gesamte System M durch die Verwendung von B entstehen.
4. Übertrage entsprechend C2 auf M.
5. Erweitere die Bedingungsmenge S um die Darstellungen für C1 und C2.

Die beschriebene Konstruktion des Zustandsübergangssystems M und der Bedingungsmenge S für ein zu prüfendes Bytecode-Programm läuft vollautomatisch ab. Das Befehlsverzeichnis V und seine Bestandteile wird, ebenso wie der Zustandsraum von M, einmal festgelegt für den bestimmten Zweck der "Bytecode-Verifikation", d.h. der Überprüfung von Sicherheitseigenschaften von Bytecode-Programmen.

Nachfolgend wird die Arbeitsweise des Modelcheckers erläutert.

Der Modelchecker prüft für jede Bedingung s aus der Bedingungsmenge S, ob sie vom Zustandsübergangssystem M erfüllt ist. Falls eine Bedingung s nicht erfüllt ist, erzeugt der Modelchecker Informationen, die zusammen mit der Kenntnis, für welche Instruktion s erzeugt wurde, verwendet werden kann, um zu analysieren, wie die Bedingungen an einen sicheren Ablauf der JVM durch das Bytecode-Programm verletzt werden können. Eine typische Ausgabe des Modelcheckers besteht aus der Angabe eines Ausführungspfades, d.h. einer Folge von M-Zuständen, der in einer Verletzung gewisser Bedingungen mündet.

Eine genaue Analyse des Ausführungspfades ist nicht notwendig. Ziel der Bytecode-Verifikation ist es lediglich festzustellen, ob eine Verletzung der Bedingungen durch das Programm möglich ist oder nicht. Informationen über das Wie können interessant sein, sind aber nicht notwendig, da lediglich gefordert ist, potentiell unsichere Programme abzuweisen. Dabei nimmt man in Kauf, auch sichere Programme abzuweisen, die zwar Bedingungen verletzen, deren genauere Analyse aber ergeben würde, daß sie in einer wirklichen Umgebung keinen Schaden anrichten könnten. Da solche Programme i. allg. nicht durch Compiler, sondern nur durch absichtsvolle Codierung entstehen, schränkt man sich nicht wesentlich ein.

Die Steuerung der Bytecode-Verifikation besteht in der Koordinierung der verschiedenen Komponenten. Eine Klassendatei wird der Bytecode-Verifikation unterzogen, indem

1. die Bytecode-Programme (Methoden) einzeln herausgelöst werden,
2. die Referenzen im Bytecode-Programm aufgelöst werden (Vorverarbeitung),
3. ein Zustandssystem und eine Bedingungsmenge konstruiert wird,
4. der Modelchecker mit dieser Eingabe gestartet wird;
5. bei Erfolg des Modelcheckers wird dies für das nächste Bytecode-Programm wiederholt, bei Mißerfolg wird gemeldet, daß die Bytecode-Verifikation für die untersuchte Klassendatei fehlgeschlagen ist.

Das erfindungsgemäße Verfahren läßt sich noch in einigen Punkten erweitern.

Von der Art der Abbildung eines Bytecode-Programms auf ein endliches Zustandssystem hängt es ab, welche Eigenschaften nachgewiesen werden können. Um etwa die Zulässigkeit

eines Bytecode-Programms festzustellen, kann die oben skizzierte Abbildung benutzt werden. Durch Wahl einer anderen Abstraktions-Abbildung ist es möglich, andere Eigenschaften nachzuweisen. Eine interessante Eigenschaft wäre etwa die Beschränkung des Ressourcenverbrauchs eines Bytecode-Programms auf ein gewisses Maß.

Der Einsatz des erfindungsgemäßen Verfahrens erlaubt es, das sicherheitskritische Konzept der Bytecode-Verifikation auf einer formalen Grundlage zu implementieren. Dadurch erreicht man die höchstmögliche Sicherheit für diesen Aspekt der Java-Technologie. Man kann erwarten, daß sich die Technik darüberhinaus auch zum Nachweis weitergehender Eigenschaften von Applets einsetzen läßt, so daß ein größerer Einsatzbereich interessant wird.

Insbesondere im Umfeld Java-fähiger Smartcards (wobei sich Java-fähig darauf bezieht, Bytecode-Programme ausführen zu können), wo die Sicherheitsanforderungen extrem hoch sind, ist diese Technik interessant. Java-fähige Smartcards erlauben es, Programme zu laden und auszuführen, und zwar auch dann, wenn sie sich bereits im Besitz des Endkunden befindet (dies ist mit herkömmlichen Smartcards nicht oder nur eingeschränkt möglich). Dabei ist es von größter Wichtigkeit, daß nur solche Programme geladen und ausgeführt werden, die die Integrität der Karte und der darauf gespeicherten Daten nicht verletzen, denn der Mißbrauch solcher Daten kann erheblichen persönlichen oder finanziellen Schaden anrichten.

Durch die beschriebene Technik kann die Sicherheit von Bytecode-Programmen garantiert und durch zusätzliche Erweiterungen eine gewisse Funktionalität zugesichert werden. Damit kann das Vertrauen in Anwendungen erhöht werden, die auf sicherheitskritischen Plattformen wie Smartcards ablaufen sollen.

Bezugszeichenliste

JVM	Interpreter für Java-Bytecode-Programme (Java Virtual Machine)
M	endliches Zustandsübergangssystem
S	Bedingungsmenge
s	Bedingung von S
V	Befehlsverzeichnis für Bytecode-Instruktionen
C1	Bedingungen, die ein Zustand der JVM erfüllen muß
C2	Bedingungen, die die JVM erfüllen muß
A1	Eigenschaft, die für Bedingungen in V gilt
B	Programminstruktion
P	Parameter für B

Patentansprüche:

1. Verfahren zur Prüfung von Java-Bytecode-Programmen auf Sicherheitseigenschaften nach dem Prinzip der Bytecode-Verifikation,

dadurch gekennzeichnet, daß

a) die Funktionsweise des zu prüfenden Bytecode-Programms unter Verwendung eines Algorithmus, der das Verhalten der Bytecode-Instruktionen beschreibt, von einem potentiell unendlichen Zustandsübergangssystem auf ein endliches Zustandsübergangssystem (M) und der Zustandsraum des Interpreters (JVM) auf eine endliche Menge von Zuständen im endlichen Zustandsübergangssystem (M) abgebildet werden, wobei alle nicht für die Zulässigkeit des zu prüfenden Bytecode-Programms erforderlichen Informationen entfallen, so daß das daraus resultierende endliche Zustandsübergan-gssystem (M) ausschließlich Typinformationen zum Nachweis der Zulässigkeit des Bytecode-Programms enthält, welche in einen Modelchecker eingegeben werden, daß

b) die Eigenschaften, die ein zulässiges Bytecode -Programm kennzeichnen, in einer Logik in Form von Formeln erfaßt und als Bedingungsmenge (S) in den Modelchecker eingegeben werden, wobei der Modelchecker jede einzelne Bedingung (s) der Bedingungsmenge (S) als Spezifikationssprache für Systemeigenschaften von Bytecode-Programmen interpretiert, und daß

der Modelchecker für jede Bedingung (s) der Bedingungsmenge (S) prüft, ob ob sie vom Zustandsübergangssystem (M) erfüllt ist, und daß das überprüfte Bytecode-Programm automatisch zur weiteren Verarbeitung freigegeben wird, wenn das Zustandsübergangssystem (M) alle Bedingungen (s) der Bedingungsmenge (S) erfüllt.

INTERNATIONAL SEARCH REPORT

Interr ıal Application No
PCT/EP 99/04438

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F11/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 685 792 A (AT&T CORP.) 6 December 1995 (1995-12-06) abstract	1

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 October 1999	28/10/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Corremans, G

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

information on patent family members

Inter nal Application No
PCT/EP 99/04438

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 685792 A	06-12-1995	CA 2147536 A	02-12-1995
		JP 7334566 A	22-12-1995
		US 5615137 A	25-03-1997

INTERNATIONALER RECHERCHENBERICHT

Intern ales Aktenzeichen
PCT/EP 99/04438

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
IPK 6 G06F11/00

Nach der Internationalen Patentklassifikation (IPK) oder nach der nationalen Klassifikation und der IPK

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)
IPK 6 G06F

Recherchierte aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	EP 0 685 792 A (AT&T CORP.) 6. Dezember 1995 (1995-12-06) Zusammenfassung -----	1

☐ Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen

☒ Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :

"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist

"E" älteres Dokument, das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist

"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)

"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht

"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist

"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist

"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden

"Y" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren anderen Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist

"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche	Absendedatum des internationalen Recherchenberichts
20. Oktober 1999	28/10/1999

Name und Postanschrift der Internationalen Recherchenbehörde	Bevollmächtigter Bediensteter
Europäisches Patentamt, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Corremans, G

1

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Intern ales Aktenzeichen.
PCT/EP 99/04438

Im Recherchenbericht angeführtes Patentdokument	Datum der Veröffentlichung	Mitglied(er) der Patentfamilie	Datum der Veröffentlichung
EP 685792 A	06-12-1995	CA 2147536 A	02-12-1995
		JP 7334566 A	22-12-1995
		US 5615137 A	25-03-1997